INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Fast Computation With Two Algebraic Numbers

Alin BOSTAN, Philippe FLAJOLET, Bruno SALVY, Éric SCHOST

N° 4579

Octobre 2002

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--4579--FR+ENG

Fast Computation With Two Algebraic Numbers

Alin BOSTAN, Philippe FLAJOLET, Bruno SALVY, Éric SCHOST

Thème 2 — Génie logiciel
et calcul symbolique

Projet Algorithmes

Rapport de recherche n° 4579 — Octobre 2002 — 20 pages

Abstract: We propose fast algorithms for computing composed multiplications and composed sums, as well as "diamond products" of univariate polynomials.

Key-words: resultant, composed sums, composed multiplications, diamond product

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex (France)
Téléphone : +33 1 39 63 55 11 — Télécopie : +33 1 39 63 53 30

Calcul rapide avec deux nombres algébriques

Résumé : Nous présentons des algorithmes rapides pour le calcul du produit composé et de la somme composée, ainsi que pour le "produit diamant" de polynômes univariés.

Mots-clés : résultant, somme composée, produit composé, produit diamant

FAST COMPUTATION WITH TWO ALGEBRAIC NUMBERS

ALIN BOSTAN, PHILIPPE FLAJOLET, BRUNO SALVY, AND ÉRIC SCHOST

Abstract. We propose fast algorithms for computing composed multiplications and composed sums, as well as "diamond products" of univariate polynomials.

1. Introduction

Let k be an effective field and let f and g be monic polynomials in $k[T]$, of degrees m and n respectively. We are interested in computing their *composed sum* $f \oplus g$ and their *composed multiplication* $f \otimes g$, which are polynomials of degree $D := mn$ defined by

$$f \oplus g = \prod_{\alpha,\beta} \big(T - (\alpha + \beta)\big) \quad \text{and} \quad f \otimes g = \prod_{\alpha,\beta} (T - \alpha\beta),$$

the products running over all the roots α of f and β of g, counted with multiplicities, in an algebraic closure $\overline{k}$ of k.

More generally, given a bivariate polynomial $H \in k[X, Y]$, of degree in X less than m and of degree in Y less than n, we study the fast computation of the so-called *diamond product* $f \diamond_H g$ of f and g, which is the polynomial of degree $D = mn$ defined by

$$f \diamond_H g = \prod_{\alpha,\beta} \big(T - H(\alpha, \beta)\big),$$

the product running over all the roots α of f and β of g, counted with multiplicities. This polynomial, *a priori* defined over $\overline{k}$, has in fact coefficients in the base field k.

The operation $\diamond_H$ was introduced by Brawley and Carlitz in [7]. They showed that if f and g are polynomials and k is finite, their diamond product enjoys the following remarkable property: $f \diamond_H g$ is irreducible if and only if both f and g are irreducible and their degrees co-prime. Consequently, the composed sums and multiplications are used for constructing irreducible polynomials of large degree over finite fields, see [7, 8, 23] and [24].

These operations, in particular composed sums and composed products, actually appear as basic subroutines in many other algorithms, including computations with algebraic numbers, symbolic summation and study of linear recurrent sequences. We present some of these applications in Section 5.

Previous complexity results. The polynomials $f \oplus g$ and $f \otimes g$ can be expressed in terms of resultants, see for instance [18]:

$$(f \oplus g)(x) = \mathrm{Res}_y(f(x - y), g(y)) \ \text{ and } \ (f \otimes g)(x) = \mathrm{Res}_y(y^m f(x/y), g(y)). \tag{1}$$

The formulas (1) already show that $f \otimes g$ and $f \oplus g$ have coefficients in k. They also provide a way of computing these polynomials. However, this is not entirely satisfactory: even using the fastest algorithms for bivariate resultants [22, 27], the resultant-based computation has complexity of order $O_{\log}(M(D)\min(m,n))$ field operations in both cases. Here $M(D)$ stands for the number of base field operations required to compute the product of two polynomials of degree D and, also, the first $D+1$ coefficients in the product of two formal power series given at precision D. The symbol $O_{\log}$ indicates the presence of logarithmic terms.

In characteristic zero, Dvornicich and Traverso suggested in [13] a different approach, based on the use of the power sums of the roots of the two polynomials. Their idea is to express the power sums of the roots of $f \oplus g$ and $f \otimes g$ in terms of those of f and g and then to make mutual conversions between the power sums and the elementary symmetric sums of the roots. These conversions are made via the Newton formulas. The complexity of their algorithm is $O(D^2)$ operations in k.

Brawley, Gao and Mills proposed in [8] several algorithmic solutions for the composed multiplication and sum over a finite field. Apart from the resultant method based on formulas (1), the most efficient is the LRS (linearly recurrent sequence) method. However this method still has quadratic complexity in the degree of the output and it works only under the assumption of an irreducible output.

In [8, Section 3], Brawley, Gao and Mills also considered the problem of the efficient computation of the general diamond product. Their algorithm only works over a finite field and under the same assumption on the irreducibility of the output. If f and g have the same degree m (so that $D = m^2$), the complexity of their algorithm has order $O(D^2 M(\sqrt{D}))$.

Our contribution. This note aims to show that one can do better, both in characteristic zero and in positive characteristic. We basically reformulate the key idea in [13], that is, to represent a polynomial by the power sums of its roots, in terms of generating series.

If the characteristic of the base field is zero or large enough, this enables us to give optimal algorithms (up to logarithmic factors) for the composed sum and multiplication. These algorithms are based on formulas expressing the power sums of the roots of $f \otimes g$ and of $f \oplus g$ in terms of those of f and g. They use mainly multiplications, inversions and exponentiations of power series, for which nearly optimal algorithms are known [9], [17, Section 13.9], [27, Section 9.1]. We give a similar algorithm in arbitrary characteristic for the computation of the composed multiplication.

We also propose a fast algorithm for computing the diamond product. The key idea of our approach is to express the linearly recurrent sequence of the power sums of the roots of $f \diamond_H g$ as the traces of H^s in the quotient algebra $k[X,Y]/(f(X), g(Y))$. The computation of these traces is based on a generalization of an algorithm of Shoup [25] to the bivariate case. The algorithm we propose improves the previously known complexity to $O(\sqrt{D}M(D) + D^2)$ operations in k.

This algorithm works under no additional assumptions if the base field has characteristic zero. Over a finite field, they work under a mild assumption, which is satisfied, for instance, if the output $f \diamond_H g$ is an irreducible polynomial.

Our results are encapsulated in the following theorem:

Theorem 1. *Let f and g be two monic polynomials in $k[T]$ of degrees m and n and let $D = mn$. Then one can compute:*

(1) *the composed product $f \otimes g$ within $O(M(D))$ operations in k, if k has characteristic zero or larger than D;*
(2) *the composed product $f \otimes g$ within $O(M(D)\log(D))$ operations, if the characteristic of k is positive and larger than all the multiplicities of the roots of $f \otimes g$;*
(3) *the composed sum $f \oplus g$ using $O(M(D))$ operations in k, if k has characteristic zero or larger than D.*

Suppose $H \in k[X, Y]$ has degree in X less than m and degree in Y less than n. Then one can compute:

4. *the diamond product $f \diamond_H g$ using $O(\sqrt{D}M(D) + D^2)$ operations in k, if k has characteristic zero or larger than D, or if the characteristic of k is positive and larger than all the multiplicities of the roots of $f \diamond_H g$.*

Note that using Fast Fourier Transform for the power series multiplication yields $O(M(D)) = O_{\log}(D)$, see [27, Section8.2]. This implies that the algorithms we propose for the composed multiplication and for the composed product are nearly optimal, i.e., their complexity is linear, up to logarithmic factors, in the degree D of the output.

The next table clarifies the state of the art on the questions addressed in this paper. The arithmetical complexities are stated in terms of the degree D of the output in the case of two polynomials f and g of equal degree $m = n$, so that $D = m^2$. The unstarred entry corresponding to $f \oplus g$ in arbitrary characteristic is obtained by the resultant computation (1). The other two entries in the case of the arbitrary characteristic are valid under the assumptions of Theorem 1.

characteristic	$f \otimes g$	$f \oplus g$	$f \diamond_H g$
zero or $> D$	$O(M(D))$ $\star$	$O(M(D))$ $\star$	$O(\sqrt{D}M(D) + D^2)$ $\star$
arbitrary	$O(M(D)\log(D))$ $\star$	$O_{\log}(\sqrt{D}M(D))$	$O(\sqrt{D}M(D) + D^2)$ $\star$

TABLE 1. A bird's eye-view on complexities; the symbol $\star$ indicates our contribution.

Outline of the paper.

- In Section 2, we propose fast algorithms for the translation between a monic polynomial and the power sums of its roots. Depending on the characteristic of the base field, we split the problem into two cases, which are detailed in Subsections 2.1 and 2.2.
- In Section 3 we use these results to compute the composed multiplication in arbitrary characteristic and the composed sum in characteristic zero or large enough, and we present the experimental behavior of our algorithms.
- In Section 4 we address the problem of computing the diamond product $f \diamond_H g$. We show that it amounts to evaluating the trace form on the successive powers of H in the quotient algebra $k[X,Y]/(f(X), g(Y))$. This is a particular instance of the power projection problem; we solve it using a generalization of a "baby-step / giant-step" algorithm of Shoup's.
- Section 5 presents several applications of these composed operations and describes two related questions: the fast computation of resolvents and of Graeffe polynomials.

2. Power Sums and Elementary Symmetric Sums of Roots

The key to our approach is to represent a polynomial by the power sums of its roots and conversely, to reconstruct the polynomial from these power sums; this idea is already present in [13]. Newton formulas provide a straightforward way to make these conversions, but they apply only in characteristic zero (or large enough) and have quadratic complexity in the degree of the polynomial. In this section, we describe optimal algorithms (i.e., linear, up to logarithmic factors, in the degree of the polynomial) for these conversions.

The translation from a polynomial to the power sums of its roots is quite simple and is based on Lemma 1 below; for the opposite conversion, we distinguish two cases. In Subsection 2.1 we treat the case of the characteristic zero or large enough, for which the solution is based on the existence of the exponential of a power series. Note that in characteristic zero a similar treatment is given in [17, Section 13.8] and in [4, Section 1.4]. In the positive arbitrary characteristic case and under some mild assumptions, our solution relies on the Berlekamp-Massey algorithm, see Subsection 2.2.

Notation. In the rest of this article, we will make use of the following notation:

- $N_s(h)$ denotes the s-th *power sum* of the roots of polynomial $h \in k[T]$, that is, the sum $\sum_\gamma \gamma^s$, taken over all the roots of h in $\overline{k}$, counted with multiplicities.
- If P is a polynomial in $k[T]$, we write $\mathrm{rev}(P)$ for its *reverse*, namely $T^{\deg(P)} P(\frac{1}{T})$.
- The *logarithmic reverse* $\mathrm{LogRev}(h)$ of $h \in k[T]$ denotes the rational power series
$$\frac{\mathrm{rev}(h')}{\mathrm{rev}(h)} \in k[[T]].$$

The first result helps us estimate the cost of conversion between power sums and elementary symmetric sums.

Lemma 1. *Let h be a polynomial in $k[T]$. Then, the following formula holds:*

$$\mathrm{LogRev}(h) = \sum_{s\geq 0} N_s(h)T^s.$$

Proof. Let $\gamma_1, \ldots, \gamma_D$ be the roots of h in $\overline{k}$. By logarithmic differentiation we have

$$\frac{\mathrm{rev}(h')}{\mathrm{rev}(h)} = \sum_{i=1}^{D} \frac{1}{1-\gamma_i T} = \sum_{i=1}^{D} \left(\sum_{s\geq 0} \gamma_i^s T^s \right) = \sum_{s\geq 0} \left(\sum_{i=1}^{D} \gamma_i^s \right) T^s.$$

□

Corollary 1. *If h is a polynomial of degree D in $k[T]$, the first N power sums of the roots of h can be computed within $O(M(\max(N, D)))$ base field operations.*

The converse direction is more difficult to handle: while in characteristic zero the Newton formulas give a one-to-one correspondence between power sums and elementary symmetric sums, in the positive characteristic case distinct monic polynomials of the same degree may have equal power sums of roots. Consequently, our treatment will take into account the characteristic of the base field.

The results of the next subsections are summarized in the following proposition.

Proposition 1. *Let h be a polynomial of degree D in $k[T]$.*

(1) *If k has characteristic zero or greater than D, then the polynomial h can be computed from the first D power sums of its roots within $O(M(D))$ base field operations.*
(2) *If k has positive characteristic p and if all the roots of h have multiplicities less than p, then the polynomial h can be computed from the first $2D$ power sums of its roots within $O(M(D)\log(D))$ base field operations.*

We stress the fact that the algorithm for positive characteristic also applies in characteristic zero, under no further assumptions on h. However, the first algorithm is more efficient in this case, since it saves a logarithmic factor.

We now focus on proving Proposition 1. In Subsection 2.1 we study the case of characteristic zero or large enough. In Subsection 2.2 we address the arbitrary positive characteristic case.

2.1. **The case of characteristic zero or large enough.** The *exponential* of a power series F with positive valuation over a field k of characteristic zero is defined by

$$\exp(F) := \sum_{s\geq 0} \frac{F^s}{s!}.$$

If the base field k has positive characteristic p, we define, as an analogue to the exponential of a power series $F \in k[[T]]$ with positive valuation, the series

$$\exp(F) := \sum_{s=0}^{p-1} \frac{F^s}{s!}.$$

The next lemma expresses the reverse of a polynomial h as the exponential of a series involving its logarithmic reverse. It translates the reciprocity between logarithm and exponential into an equality between power series.

Lemma 2. *Let h be a monic polynomial of degree D in $k[T]$, where k is a field of characteristic zero or larger than D. Then the following formula holds:*

$$\mathrm{rev}(h) = \exp\left(\int \frac{1}{T}\cdot\big(D - \mathrm{LogRev}(h)\big)\right).$$

Proof. Let $\gamma_1, \ldots, \gamma_D$ be the roots of h in $\overline{k}$. From Lemma 1, it follows immediately that:

$$\frac{1}{T}\cdot\Big(D - \mathrm{LogRev}(h)\Big) = -\sum_{s\geq 0} N_{s+1}(h)T^s.$$

On the other hand, a similar calculation shows that:

$$\frac{\mathrm{rev}(h)'}{\mathrm{rev}(h)} = \sum_i \frac{-\gamma_i}{1-\gamma_i T} = -\sum_{s\geq 0}\Big(\sum_i \gamma_i^{s+1}\Big)T^s.$$

As one can easily verify, for a polynomial $P \in k[T]$ with constant coefficient 1, the formula $P = \exp(\int P'/P)$ holds as soon as k has characteristic zero or larger than the degree of P. By applying this fact to $P = \mathrm{rev}(h)$ in conjunction with the previous two formulas, we conclude the proof of the lemma. □

Corollary 2. *A monic polynomial h of degree D over a field of characteristic zero or larger than D can be computed from the first D power sums of its roots within $O(M(D))$ base field operations.*

Proof. (Compare [4, p. 34–35]) By assumption, we know the series $\mathrm{LogRev}(h)$ at precision D, from which we deduce $\int \frac{1}{T}\cdot\big[D - \mathrm{LogRev}(h)\big]$ at precision D in linear time.

Then only the first D coefficients of the exponential can be nonzero. These coefficients can be computed within $O(M(D))$ field operations, using a Newton iteration; see [9]. Finally, the polynomial

$$h = \mathrm{rev}(\mathrm{rev}(h))\cdot T^{D-\deg(\mathrm{rev}(h))}$$

can be recovered in linear complexity. This concludes the proof of the corollary. □

This proves the first part of Proposition 1.

2.2. **The arbitrary positive characteristic case.** We now prove the second part of Proposition 1. The key ingredient is the following lemma:

Lemma 3. *Let h be a polynomial in $k[T]$ and $\big(N_s(h)\big)_{s\geq 0}$ the sequence of the power sums of its roots. Then, this sequence is linearly recurrent; its minimal polynomial is*

$$\frac{h}{\gcd(h, h')}.$$

Proof. By Lemma 1, the generating series of the sequence $\big(N_s(h)\big)_{s\geq 0}$ is rational, so this sequence is linearly recurrent. By [16, Lemma 1], its minimal polynomial is given by the denominator of the irreducible form of the rational series

$$\sum_{s\geq 0} \frac{N_s(h)}{T^{s+1}}.$$

On the other hand, if γ_i denote the roots of h in $\overline{k}$, the previous series can be written

$$\sum_{i=1}^{D} \Big(\sum_{s\geq 0} \frac{\gamma_i^s}{T^{s+1}} \Big) = \sum_{i=1}^{D} \frac{1}{T-\gamma_i} = \frac{h'}{h},$$

so the denominator of its irreducible form is $\frac{h}{\gcd(h,h')}$. This concludes the proof. □

Corollary 3. *Let h be a monic polynomial of degree D over a field of characteristic p. Suppose that all the roots of h have multiplicities less than p. Then h can be computed from the first $2D$ power sums of its roots within $O(M(D)\log(D))$ operations in k.*

Proof. We first show that the algorithm below correctly outputs a polynomial given its logarithmic reverse. For a rational series $S = \sum_{i\geq 0} s_i T^i$ in $k[[T]]$, we denote by `MinimalPolynomial`(S) the minimal polynomial of the linearly recurrent sequence $(s_i)_{i\geq 0}$ of its coefficients.

Recovering a polynomial from its logarithmic reverse

Input: the first $2D$ terms of the series $\mathrm{LogRev}(h)$.
Output: the polynomial h.

$S_1 \leftarrow \mathrm{LogRev}(h) + O(T^{2D})$;
$i \leftarrow 1$;
repeat
  $v_i \leftarrow$ `MinimalPolynomial`(S_i);
  $D_i \leftarrow \deg(v_i)$;
  $R_i \leftarrow \mathrm{LogRev}(v_i) + O(T^{2D_i})$;
  $S_{i+1} \leftarrow S_i - R_i$;
  $i \leftarrow i+1$;
until $D_i = 0$;
$t \leftarrow i-1$;
$v \leftarrow v_1 v_2 \cdots v_t$;
return v.

Let $h = \prod_\gamma (T-\gamma)^{\mu(\gamma)}$ be the factorization of h over an algebraic closure of k, where $\mu(\gamma)$ denotes the multiplicity of the root γ. Since h is defined over a field of

characteristic p we have that

$$\frac{h}{\gcd(h', h)} = \prod_{p \not| \mu(\gamma)} (T - \gamma),$$

and by the assumption on the multiplicities $\mu(\gamma)$, the product above is taken over *all* the roots of h. Using Lemma 3, the minimal polynomial v_1 of the sums sequence of h will thus equal $\prod_\gamma (T - \gamma)$. Since $\mathrm{LogRev}(AB) = \mathrm{LogRev}(A) + \mathrm{LogRev}(B)$, the series $S_2 = \mathrm{LogRev}(h) - \mathrm{LogRev}(v_1)$ is the logarithmic reverse of

$$\gcd(h, h') = \prod_\gamma (T - \gamma)^{\mu(\gamma)-1}.$$

Therefore, using Lemma 3 again, v_2 equals the product of the linear factors $(T - \gamma)$ taken over the set of the roots of h of multiplicity at least 2. Iterating this argument, we obtain, for all $1 \leq i \leq t$ the formula

$$v_i = \prod_{\{\gamma \mid \mu(\gamma) \geq i\}} (T - \gamma).$$

This shows that h is exactly the product of the square-free polynomials v_i.

Let us now estimate the cost of our algorithm. The minimal polynomial computation in the loop i can be done using the Berlekamp-Massey algorithm, which has complexity $O(M(D_i)\log(D_i))$, see [27, Ch. 11,12]. Moreover, by Lemma 2, computing R_i requires $O(M(D_i))$ operations in k. Using the sub-additivity of M,

$$\sum_{i=1}^{t} M(D_i)\log(D) \leq M\Big(\sum_{i=1}^{t} D_i\Big)\log(D) = M(D)\log(D),$$

this yields a total cost for the iterative steps of $O(M(D)\log(D))$ base field operations.

Finally, the product of all the intermediate polynomials v_i can be computed using at most $M(D)\log(t)$ operations, see [27, Algorithm 10.3]. This is also in $O(M(D)\log(D))$, since $t \leq D$. This concludes the proof of Proposition 1. □

Square-free decomposition. Using the algorithm described in the proof above, we can actually compute, within the same cost, a square free decomposition $(h_i)_i$ of h. To do this, it is sufficient to compute the successive quotients $h_i = v_i/v_{i+1}$. Classically, the square-free decomposition of h is achieved by an algorithm of Yun [27, Algorithm 14.21] within the same complexity as ours, but Yun's algorithm takes the polynomial h as input. It turns out that these two algorithms are closely related, the successive gcd computations in Yun's algorithm being easy to translate in terms of our LogRev series.

As in Yun's algorithm, if the assumption of Proposition 2 is violated, our algorithm returns a proper divisor of the desired output, so we can *a posteriori* detect whether h has roots of multiplicities greater than or equal to p.

3. Two Useful Resultants That Can Be Computed Fast

The results of the preceding section will help us design optimal algorithms for the computation of the composed multiplication $f \otimes g$ and composed sum $f \oplus g$ of two

monic polynomials f and g of degrees m and n. These are particular instances of resultants

$$(f \otimes g)(x) = \prod_{g(\beta)=0} \beta^m f(x/\beta) = \mathrm{Res}_y\big(y^m f(x/y), g(y)\big),$$

and

$$(f \oplus g)(x) = \prod_{g(\beta)=0} f(x-\beta) = \mathrm{Res}_y\big(f(x-y), g(y)\big)$$

that can be computed faster than the general bivariate resultants, for which no optimal algorithm is known [27].

Our algorithms are based on formulas expressing the logarithmic reverses of $f \otimes g$ and of $f \oplus g$ in terms of those of f and g. Corollary 4 and 5 below provide the proof of points 1, 2 and 3 in Theorem 1.

3.1. **Computing the composed multiplication.** Our algorithm for the composed multiplication $f \otimes g$ is based on the following lemma:

Lemma 4. *Let f and g be two polynomials in $k[T]$. Then, the following formula holds:*

$$\mathrm{LogRev}(f \otimes g) = \mathrm{LogRev}(f) \odot \mathrm{LogRev}(g),$$

where $\odot$ denotes the Hadamard product (that is, the termwise product) of power series.

Proof. For $s \geq 0$, the s-th power sum of the roots of $f \otimes g$ is $\sum_{\alpha,\beta}(\alpha\beta)^s$, the sum running over all the roots α of f and β of g. This sum can be rewritten as $\big(\sum_\alpha \alpha^s\big) \cdot \big(\sum_\beta \beta^s\big)$, which is the product of the s-th power sums of the roots of f and of g. This proves that the series $\mathrm{LogRev}(f \otimes g)$ is the Hadamard product of $\mathrm{LogRev}(f)$ and $\mathrm{LogRev}(g)$. □

Corollary 4. *Let f and g be two monic polynomials of degrees m and n over a field k and let $D = mn$. Then the composed multiplication $f \otimes g$ can be computed using*

(1) *$O(M(D))$ operations in k, if the characteristic of k is zero or greater than D;*
(2) *$O(M(D)\log(D))$ operations in k if the characteristic of k is positive and greater than all the multiplicities of the roots of $f \otimes g$.*

Proof. We apply Lemma 1, Proposition 1 and Lemma 4 and the fact that computing the Hadamard product of two series at precision D is linear in D. □

3.2. **Computing the composed sum in characteristic zero or large enough.** Let k be a field of arbitrary characteristic and let $\mathrm{E} \in k[[T]]$ denote the power series

$$\mathrm{E} = \exp(T),$$

where exp denotes the exponential defined in Section 2.1. Then the algorithm for $f \oplus g$ is based on the following lemma:

Lemma 5. *Let f and g be two polynomials in $k[T]$. Then*

(1) *if the characteristic of k is zero, the following formula holds:*

$$\mathrm{LogRev}(f \oplus g) \odot \mathrm{E} = \big(\mathrm{LogRev}(f) \odot \mathrm{E}\big) \cdot \big(\mathrm{LogRev}(g) \odot \mathrm{E}\big);$$

(2) *if $p > 0$ is the characteristic of k, the following formula holds:*

$$\mathrm{LogRev}(f \oplus g) \odot \mathrm{E} = \big(\mathrm{LogRev}(f) \odot \mathrm{E}\big) \cdot \big(\mathrm{LogRev}(g) \odot \mathrm{E}\big) \mod T^p.$$

Proof. We give the proof in the zero characteristic case; the same arguments apply *mutatis mutandis* in the second case, by judiciously truncating the series involved.

The definition of LogRev implies that

$$\mathrm{LogRev}(f \oplus g) = \sum_{s \geq 0} \left(\sum_{\alpha,\beta} (\alpha + \beta)^s \right) T^s,$$

the second sum running over all the roots α of f and β of g. The conclusion now reads

$$\sum_{s \geq 0} \frac{\sum_{i,j} (\alpha + \beta)^s}{s!} T^s = \left(\sum_{s \geq 0} \frac{\sum_{\alpha} \alpha^s}{s!} T^s \right) \cdot \left(\sum_{s \geq 0} \frac{\sum_{\beta} \beta^s}{s!} T^s \right)$$

and we are done, as the latter equality simply translates the fact that

$$\sum_{\alpha,\beta} \exp\big((\alpha + \beta)T\big) = \Big(\sum_{\alpha} \exp\big(\alpha T\big) \Big) \cdot \Big(\sum_{\beta} \exp\big(\beta T\big) \Big).$$

□

Corollary 5. *Let f and g be two polynomials of degrees m and n over a field k of characteristic zero or greater than $D = mn$. Then the composed sum $f \oplus g$ can be computed using $O(M(D))$ operations in k.*

Proof. Once again, this is a direct application of Lemma 1, Proposition 1 and Lemma 5. □

Experimental Results. We have implemented our algorithm for the composed sum in the Magma computer algebra system [5]. This choice was motivated by the fact that, to the best of our knowledge, Magma is the only general purpose computer algebra system in which implementations of the fastest algorithms for power series multiplication are available.

We compare the timings given by Magma's `Resultant` computation to those provided by our algorithm. We stress the fact that, as far as we know, the algorithm implemented in Magma for the resultant has quadratic complexity in the degree of the output.

Since the complexity estimates are stated in terms of number of operations in the base field, we choose to experiment on the finite field with $10^{30} + 57$ elements. The polynomials have equal degrees $m = n$ and their coefficients are chosen randomly; the output has degree $D = m^2$.

In Figure 1 we draw the complexity curves corresponding both to our algorithm and to the resultant-based algorithm. The degree of the output polynomial is given on the horizontal axis; on the vertical axis the time spent by the two algorithms

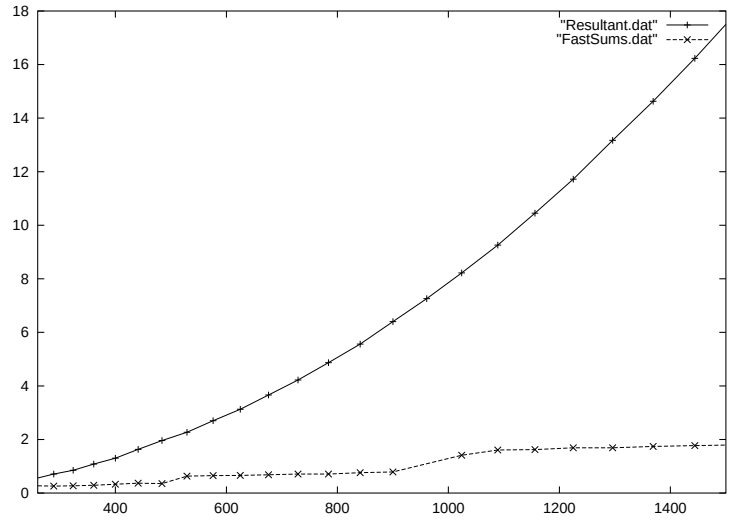

FIGURE 1. Our algorithm versus the resultant computation. (Time in sec. vs degree)

is given in seconds. Table 2 gives some running times of our algorithm on input polynomials for which the resultant computation takes more than 1 h of CPU time.

All the tests were made on the computers of the MEDICIS resource center[1] using a 1.5 GB, 1 Ghz AMD Athlon processor.

$m = n$	150	200	250	300
D	22500	40000	62500	90000
`Time`	79.91	229.95	244.03	543.71

TABLE 2. Computation of the composed sum; time is given in seconds.

4. Computing the Diamond Product

We finally address the general case: computing the *diamond product* of f and g. From the data of a polynomial $H(X, Y)$, of degree less than m in X and n in Y, the diamond product is defined as the polynomial of degree $D = mn$

$$f \diamond_H g = \prod_{\alpha,\beta} \big(T - H(\alpha, \beta)\big),$$

where the product runs over all the roots α of f and β of g, counted with multiplicities. In this section, we prove that computing $f \diamond_H g$ can be achieved within

$$O(\sqrt{D}M(D) + D^2)$$

[1] http://www.medicis.polytechnique.fr

operations in k.

4.1. **Computations in the quotient algebra.** Let Q be the quotient algebra $k[X,Y]/(f(X),g(Y))$. In the rest of this section, we repeatedly use the *trace*, which is a linear form defined on Q: the trace of $A \in Q$ is defined as the trace of the endomorphism of multiplication by A in Q.

Our algorithm is based on the following fundamental fact, which is sometimes referred to as Stickelberger's Theorem, see [12, Proposition 2.7]: for any A in Q, the characteristic polynomial of A equals $\prod_{\alpha,\beta}\big(T - A(\alpha,\beta)\big)$, where the product runs over all the roots of f and g counted with multiplicities. As a corollary, we have the following lemma:

Lemma 6. *The polynomial $f \diamond_H g$ is the characteristic polynomial of H in Q. The s-th power sum of the roots of $f \diamond_H g$ is the trace of H^s in Q.*

A first consequence of this result is that the coefficients of $f \diamond_H g$ are in k. Also by Lemma 6, proving the final part of Theorem 1 amounts to giving a fast computation scheme for the first traces of H^s in Q. This is the object of the following proposition.

Proposition 2. *Given $N \geq 0$, the sequence*

$$\mathrm{trace}(H), \mathrm{trace}(H^2), \ldots, \mathrm{trace}(H^N)$$

can be computed within $O(\sqrt{N}M(D) + ND)$ base field operations.

We immediately deduce the proof of the complexity estimates in Theorem 1: from Proposition 1, the number of traces to be computed is at most $2D$. Using the above proposition, this has complexity $O(\sqrt{D}M(D) + D^2)$. Then Proposition 1 shows that the additional cost of recovering $f \diamond_H g$ from the power sums of its roots is negligible.

Thus, we now concentrate on proving Proposition 2.

4.2. **Power projection.** Computing the traces of the first N powers of H is a particular instance of the *power projection* problem: given a linear form ℓ on the k-algebra Q, compute the sequence $[\ell(1), \ell(H), \ldots, \ell(H^N)]$.

The naïve method consists in computing all powers of H, and then applying ℓ to all of them. Anticipating the results of the following paragraphs, we note that this method has complexity $O(NM(D))$. We now present a faster solution, based on ideas developed by Shoup in [23, 25], (see also [6] for another application of these ideas to the context of polynomial system solving). The dual $\widehat{Q}$ is endowed with a natural Q-module structure: given $A \in Q$ and a linear form $\ell \in \widehat{Q}$, one defines the *transposed product* of (A,ℓ) as the following linear form:

$$\begin{array}{rccl} A \circ \ell : & Q & \to & k \\ & B & \mapsto & \ell(AB). \end{array}$$

The denomination "transposed product" expresses that the map $\ell \mapsto A \circ \ell$ is the transposed of the usual product $B \mapsto AB$.

Using this module structure on the dual space, Shoup proposed in [25] the following "baby step/giant step" algorithm for the power projection, which avoids the computation of all the powers of H:

Power projection

Input: H in Q, ℓ in $\widehat{Q}$, N in $\mathbb{N}$.
Output: the sequence $[\ell(1), \ell(H), \ldots, \ell(H^N)]$.

$t \leftarrow \lfloor \sqrt{N} \rfloor, t' \leftarrow \lceil N/t \rceil$.
$H_q \leftarrow H^q, \ \ q = 0, \ldots, t$
`for` $q \leftarrow 0, \ldots, t' - 1$ `do`
    $c_{qt+r} \leftarrow \ell(H_r), \ \ r = 0, \ldots, t-1$
    $\ell \leftarrow H_t \circ \ell$
`return` $[c_0, \ldots, c_N]$;

A quick look at this algorithm shows that it requires N evaluations of a linear form, $O(\sqrt{N})$ multiplications in Q and $O(\sqrt{N})$ transposed multiplications.

We now proceed to inspect the cost of each of these steps: in the following subsections, we prove that the evaluation of a linear form has complexity D, and that product and transposed product have complexity $O(M(D))$. This provides the proof of Proposition 2.

4.3. **Representing the linear forms.** The quotient algebra Q has a canonical monomial basis: since f has degree m and g has degree n, then

$$\mathcal{M} = \{x^i y^j, 0 \leq i < m, 0 \leq j < n\}$$

forms a monomial basis of Q, where x and y are the images of X and Y in Q.

The linear forms will be given by their coefficients in the dual basis of $\mathcal{M}$, that is, by the list of their values on the elements in $\mathcal{M}$. Then the cost of a single evaluation is $mn = D$ operations in the base field.

As a preamble to our algorithm, it is also necessary to compute the trace of all elements in the basis $\mathcal{M}$.

Let us thus consider i in $0, \ldots, m-1$ and j in $0, \ldots, n-1$. By Stickelberger's theorem, the trace of $x^i y^j$ is $\sum_{\alpha,\beta} \alpha^i \beta^j$; then Lemma 4 shows that this trace is the product of the coefficients of T^i in $\mathrm{LogRev}(f)$ and T^j in $\mathrm{LogRev}(g)$.

The series $\mathrm{LogRev}(f)$ and $\mathrm{LogRev}(g)$ can be computed at precision respectively m and n in $O(M(\max(m,n)))$ base field operations. Then by the above reasoning, the value of the trace form on the canonical basis $\mathcal{M}$ can be computed for $mn = D$ additional multiplications.

4.4. **Complexity of the product in Q.** Due to the very specific form of the ideal defining Q, we can design a simple and fast algorithm for the multiplication in Q, with complexity $O(M(D))$.

Let A, B in Q. To obtain AB in Q, we first compute their product as plain polynomials in $k[X,Y]$, then reduce this product modulo $(f(X), g(Y))$.

Using Kronecker's substitution [27, Section 8.4], the multiplication of A, B as polynomials in $k[X,Y]$ reduces to univariate multiplication of polynomials of degree $O(mn) = O(D)$, so it can be done with complexity $O(M(D))$. The product $C = AB$ is a polynomial of degree at most $2m$ in X and $2n$ in Y; we now proceed to reduce it modulo $(f(X), g(Y))$.

We first consider C as a polynomial in $k[X][Y]$; then the reduction modulo $f(X)$ simply consists in reducing all its coefficients modulo $f(X)$. This can be done within $O(nM(m))$ operations in k; we obtain a polynomial D, which we now see in $k[Y][X]$. The second and final step, the reduction modulo $g(Y)$, is quite similar: it consists in reducing all coefficients modulo $g(Y)$, which can be done within $O(mM(n))$ operations in k.

As both $mM(n)$ and $nM(m)$ are in the class $O(M(mn)) = O(M(D))$, our algorithm for the product in Q uses $O(M(D))$ operations in k.

4.5. **Complexity of the transposed product.** We finally address the question of the complexity of the transposed multiplication in Q: we prove that it has complexity $O(M(D))$, just like the plain multiplication.

A general principle, known as the *transposition principle* [20], already asserts that these two questions have the same complexity. Our solution does not use the transposition principle, but sheds more light on the operations associated to the transposed product.

Given a linear form ℓ in $\widehat{Q}$, let us consider the bivariate generating series

$$S(\ell) = \sum_{i \geq 0, j \geq 0} \ell(x^i y^j) X^i Y^j.$$

In [6, Proposition 1], it is proved that the series $S(\ell)$ is rational, and can be written

$$S(\ell) = \frac{N(\ell)}{\mathrm{rev}(f)\mathrm{rev}(g)},$$

where $N(\ell)$ is a polynomial in $k[X,Y]$ of degree less than m in X and less than n in Y. This representation underlies our algorithm for the transposed product.

Let A be in $k[X,Y]$, with degree in X less than m and degree in Y less than n. We define $\mathrm{REV}_{m,n}(A)$, the *reverse* of A, as follows: assume that $A = \sum A_{i,j} x^i y^j$, for $i < m, j < n$; then we write $\mathrm{REV}_{m,n}(A) = \sum A_{i,j} x^{m-i} y^{n-j}$. With this notation, the following proposition shows how the numerator $N(\ell)$ behaves under transposed multiplication.

Proposition 3. *Let ℓ be in $\widehat{Q}$ and A in Q. Then the following formula holds:*

$$\mathrm{REV}_{m,n}(N(A \circ \ell)) = A \cdot \mathrm{REV}_{m,n}(N(\ell)) \mod (f(X), g(Y)).$$

Before proving this proposition, let us see how it answers the question of the complexity of the transposed product.

Corollary 6. *Let ℓ be in $\widehat{Q}$ and A in Q. Then the product $A \circ \ell$ can be computed in time $O(M(D))$.*

Proof. Knowing the values of ℓ on the canonical basis $\mathcal{M}$ amounts to knowing the restriction of $S(\ell)$ to all monomials in $\mathcal{M}$. Thus we have enough information to

recover $N(\ell)$, through the multiplication by $\mathrm{rev}(f)\mathrm{rev}(g)$, within $O(M(D))$ operations.

We deduce $\mathrm{REV}_{m,n}(N(\ell))$ by rearranging the terms of $N(\ell)$ in linear complexity. From this we obtain $\mathrm{REV}_{m,n}(N(A\circ\ell))$ using the proposition above; then we switch back to $N(A\circ\ell)$ by another reversion. We finally recover the values of $A\circ\ell$ on the monomial basis through the division by $\mathrm{rev}(f)\mathrm{rev}(g)$ in the power series ring $k[[X,Y]]$.

We proved in the previous subsection that the cost of a product modulo the ideal $(f(X),g(Y))$ is in $O(M(D))$. This gives the complexity estimate. □

This last corollary ends the complexity analysis of Proposition 2. We finally turn to the proof of Proposition 3.

By linearity, it is enough to prove the proposition when A is the monomial $x^a y^b$, with $a<m$ and $b<n$.

We decompose the sum $S(\ell)$ as follows:

$$\begin{aligned} S(\ell) &= \sum_{i\geq 0, j\geq 0} \ell(x^i y^j) X^i Y^j \\ &= \sum_{i\geq 0, 0\leq j<b} \ell(x^i y^j) X^i Y^j + \sum_{0\leq i<a, j\geq 0} \ell(x^i y^j) X^i Y^j \\ &\quad - \sum_{0\leq i<a, 0\leq j<b} \ell(x^i y^j) X^i Y^j + \sum_{i\geq a, j\geq b} \ell(x^i y^j) X^i Y^j \\ &= S_1 + S_2 + S_3 + S_4 \end{aligned}$$

We study each summand in turn, starting with S_1.

We write S_1 as a finite sum over j of the terms $Y^j \sum_{i\geq 0}(y^j\circ\ell)(x^i)X^i$. Using [6, Proposition 1] again, we deduce that this series can be written $Y^j n_j/\mathrm{rev}(f)$, with n_j a polynomial in $k[X]$ of degree less than m. Thus S_1 can be written $N_1/\mathrm{rev}(f)$, where N_1 is a polynomial in $k[X,Y]$ of degree less than b in Y and less than m in X.

Similarly, S_2 can be written $N_2/\mathrm{rev}(g)$, where N_2 is a polynomial in $k[X,Y]$ of degree less than n in Y and less than a in X. We keep S_3 as a finite sum of monomials, and S_4 is $X^a Y^b S(A\circ\ell)$.

We multiply both sides of the above equality by $\mathrm{rev}(f)\mathrm{rev}(g)$; this yields

$$N(\ell) = N_1\mathrm{rev}(f) + N_2\mathrm{rev}(g) + S_3\mathrm{rev}(f)\mathrm{rev}(g) + X^a Y^b N(A\circ\ell)$$

We now switch to the reverse of this equality: we substitute (X,Y) by $(1/X,1/Y)$, and multiply both sides by $X^m Y^n$. We obtain

$$\begin{aligned} \mathrm{REV}_{m,n}(N(\ell)) &= N_1\left(\frac{1}{X},\frac{1}{Y}\right) Y^n f + N_2\left(\frac{1}{X},\frac{1}{Y}\right) X^m g \\ &\quad + S_3\left(\frac{1}{X},\frac{1}{Y}\right) fg + \frac{1}{X^a Y^b}\mathrm{REV}_{m,n}(N(A\circ\ell)) \end{aligned}$$

Multiplying by X^aY^b yields the final form of this equality

$$\begin{aligned} X^aY^b\mathrm{REV}_{m,n}(N(\ell)) &= N_1\left(\frac{1}{X},\frac{1}{Y}\right)X^aY^{n+b}f + N_2\left(\frac{1}{X},\frac{1}{Y}\right)X^{m+a}Y^bg \\ &\quad + S_3\left(\frac{1}{X},\frac{1}{Y}\right)X^aY^bfg + \mathrm{REV}_{m,n}(N(A\circ\ell)) \end{aligned}$$

The degree constraints of N_1, N_2, S_3 then show that all summands are actually polynomials in (X, Y), and thus $X^aY^b\mathrm{REV}_{m,n}(N(\ell)) - \mathrm{REV}_{m,n}(N(A\circ\ell))$ is in the ideal $(f(X), g(Y))$. The degree constraints on $N(A\circ\ell)$ finally imply that $\mathrm{REV}_{m,n}(N(A\circ\ell))$ is the normal form of $X^aY^b\mathrm{REV}_{m,n}(N(\ell))$ modulo $(f(X), g(Y))$. □

5. Applications and Related Questions

To conclude this article, we present several situations where composed operations, notably composed sums and products, are used. We also mention two questions somehow similar to composed operations, but for which no optimal algorithms are known to us.

5.1. **Applications.**

Algebraic numbers. Algebraically, one may represent an algebraic number by its minimal polynomial (to distinguish between conjugates, one can use numerical approximates).

If α and β are two algebraic numbers represented by their minimal polynomials f and g, the sum $\alpha+\beta$ is represented by one of the irreducible factors of the composed sum $f \oplus g$. Similarly, the product $\alpha\beta$ is represented by one of the irreducible factors of the composed multiplication $f \otimes g$. Thus the resultant methods described in [11, 18] should be replaced by our faster solutions.

The algorithms descibed in this paper also adapt to operations over algebraic functions and series as it suffices to operate with a base field of the form $k(z)$. As a matter of fact, formal ideas similar to the ones developed in Section 3 prove useful in determining the generating series of walks over the half-line determined by a fixed finite set of allowed jumps; see the "Platypus Algorithm" [sic] and the discussion of [2, p. 56–58]. (There the problem is to calculate the minimal polynomial satisfied by a product $\alpha_1 \cdots \alpha_k$ of k distinct branches of an algebraic function defined by $P(z, \alpha) = 0$.)

Gosper-Petkovšek normal forms for rational functions. In many algorithms for symbolic summation (e.g., [15], [21] and [19]) one has to solve a linear first order difference equation, the *key equation*. For example, Gosper's algorithm for hypergeometric indefinite summation [15] reduces the search for a *hypergeometric* solution f of a difference equation $f(x+1) - f(x) = g(x)$ with hypergeometric right hand side g to the search of a *polynomial* solution of an auxiliary equation of a similar form.

The key routine of this algorithm is the computation of the so-called Gosper-Petkovšek normal form for rational functions, see for example [27, Section 23.4].

Apart from gcd and shift computations, this requires the computation of a polynomial whose roots are the differences between the roots of two given polynomials. This is again a resultant of the particular form discussed in Section 3.

Computing irreducible polynomials. Constructing irreducible polynomials of prescribed degree over finite fields is a difficult and useful task. It is used, for instance, to implement arithmetic in extension fields. The most efficient algorithm is due to Shoup [23, 24]: it consists in first constructing irreducible polynomials of prime power degree, then combining them to form an irreducible polynomial.

In [24], the second step is achieved by a minimal polynomial computation, which has complexity $d^{(\omega+1)/2}$, where d is the degree of the output and $2 \leq \omega < 3$ is the linear algebra exponent. Using our algorithms for the composed multiplication or for the composed sum, the cost of this step becomes linear in d, up to logarithmic factors.

Linear recurrent sequences with infinitely many zeros. A classical result [3] says that a linear recurrent sequence has infinitely many zero terms if its minimal polynomial f has a unitary pair, that is, if it has two roots whose ratio is a root of unity.

In [28], Yokoyama, Li and Nemes give algorithms to test this condition, and if so, to find the order of the multiplicative group generated by the corresponding roots of unity. The most time-consuming part of their algorithm is the computation of a polynomial whose roots are the ratios of all pairs of roots of f. This directly reduces to the computation of a composed product.

Shift of polynomials. In [26], six algorithms for computing shifts of polynomials are proposed and their complexity is analyzed. A seventh algorithm can be deduced as a straightforward application of our algorithm for the composed sums, since $f \oplus (T + a)$ is the shift polynomial of f by a. In characteristic zero, the complexity of this algorithm is linear (up to logarithmic factors) in the degree of f, in terms of base field operations. Yet, the convolution method of Aho, Steiglitz and Ullman [1] is better by a constant factor.

5.2. **Related questions and open problems.**

Resolvents. Resolvents are an important tool in Galois theory, notably for the *direct problem* of determining the Galois group of an irreducible polynomial f of degree m. Their factorization patterns help determine the Galois group of f.

For $h \leq m$, an example of such a resolvent is the polynomial f^{+h} of degree $N = \binom{m}{h}$, whose roots are the sums $\alpha_{i_1} + \cdots + \alpha_{i_h}$, with $1 \leq i_1 < \cdots < i_h \leq m$, where $(\alpha_i)_{1 \leq i \leq m}$ are the roots of f. This differs from the h-th iterated composed sum, since repetitions of roots are not allowed here. Yet, the methods we presented can help answer some simple cases, as illustrated in the following example.

Let $f(T) = T^7 - 7T + 3$ be the Cartier polynomial introduced in [14], and $F = f^{+3}$ the polynomial whose roots are all sums of $h = 3$ distinct roots of f. To prove that the Galois group of f is not the symmetric group S_7, it is enough to check that

F is not irreducible. The polynomial F has degree 35, so knowing its logarithmic reverse to order 35 suffices to recover it. The formula

$$f \oplus f \oplus f = \prod_{\alpha} (T - 3\alpha) \cdot \prod_{\alpha \neq \beta} (T - (\alpha + 2\beta))^3 \cdot \prod_{\alpha \neq \beta \neq \gamma \neq \alpha} (T - (\alpha + \beta + \gamma))^6,$$

enables us to express $\mathrm{LogRev}(F)$ as

$$\frac{1}{6} \left(\mathrm{LogRev}\left(f^{\oplus 3}\right) + 2\mathrm{LogRev}(f \otimes (T-3)) - 3\mathrm{LogRev}(f \oplus (f \otimes (T-2))) \right).$$

Using Lemmas 4 and 5, the last series can be computed from the series $\mathrm{LogRev}(f)$ and $\exp(T)$ to order 35. The polynomial F is then recovered from its logarithmic reverse, using the algorithms in Section 2. The CPU time used in the whole computation is about 300 faster than a direct resultant computation.

A straightforward generalization of this approach for an arbitrary h is not satisfactory, due to the combinatorial explosion of the number of terms involved. A faster method is presented in [10] and has complexity $O_{\log}\left(h^2 N + N^2\right)$. It is based on the following recurrence relation, expressing f^{+h} in terms of f^{+j}, for $j < h$

$$\left(f^{+h}\right)^h = \prod_{i=1}^{h} \left(\left(f \otimes (T-i)\right) \oplus f^{+(h-i)} \right)^{(-1)^{i+1}}.$$

Using this formula and the fast conversion algorithms presented in Section 2, the complexity reduces to $O_{\log}(hN)$. Nevertheless, the degree of the output is N, so an optimal algorithm for this question has yet to be found.

Graeffe polynomials. Let f be a monic polynomial of degree m and N be a positive integer. We call N-th *Graeffe polynomial* of f the polynomial whose roots are the N-th powers of the roots of f; note that it has degree m.

This polynomial can be obtained using $O(M(mN))$ operations in k, by computing the composed product of f and $X^N - 1$. Note that the same complexity is announced in [17, Section 13.8]. This complexity is almost optimal with respect to m, but not to N. On the other hand, the N-th Graeffe polynomial of f is the characteristic polynomial of X^N modulo f. Computing $X^N \mod f$ has complexity $O(M(m)\log(N))$, which is optimal in N, but then the characteristic polynomial computation has complexity more than linear in m.

Is there a way of reducing the whole cost to $O(M(m)\log(N))$? If N is a power of 2, this can be achieved using binary powering, but the general case remains open.

Acknowledgements. This work has been partly supported by the European Commission under the Future and Emerging Technologies programme of the Fifth Framework, ALCOM-FT project number IST-1999-14186.

REFERENCES

[1] A. V. Aho, K. Steiglitz, and J. D. Ullman. Evaluating polynomials at fixed sets of points. *SIAM J. Comput.*, 4(4):533–539, 1975.

[2] C. Banderier and P. Flajolet. Basic analytic combinatorics of directed lattice paths. *Theoretical Computer Science*, 281(1-2):37–80, 2002.

[3] J. Berstel and M. Mignotte. Deux propriétés décidables des suites récurrentes linéaires. *Bull. Soc. Math. France*, 104(2):175–184, 1976.

[4] D. Bini and V. Y. Pan. *Polynomial and matrix computations. Vol. 1.* Birkhäuser Boston Inc., Boston, MA, 1994. Fundamental algorithms.

[5] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *Journal of Symbolic Computation*, 24(3-4):235–265, 1997. See also `http://www.maths.usyd.edu.au:8000/u/magma/`.

[6] A. Bostan, B. Salvy, and É. Schost. Fast algorithms for zero-dimensional polynomial systems using duality. Research Report 4291, Institut National de Recherche en Informatique et en Automatique, oct 2001.

[7] J. V. Brawley and L. Carlitz. Irreducibles and the composed product for polynomials over a finite field. *Discrete Math.*, 65(2):115–139, 1987.

[8] J. V. Brawley, S. Gao, and D. Mills. Computing composed products of polynomials. In *Finite fields: theory, applications, and algorithms (Waterloo, ON, 1997)*, pages 1–15. Amer. Math. Soc., Providence, RI, 1999.

[9] R. P. Brent. Multiple-precision zero-finding methods and the complexity of elementary function evaluation. *Analytic Computational Complexity*, 1975.

[10] D. Casperson and J. McKay. Symmetric functions, m-sets, and Galois groups. *Math. Comp.*, 63(208):749–757, 1994.

[11] H. Cohen. *A course in computational algebraic number theory.* Springer-Verlag, Berlin, 1993.

[12] D. Cox, J. Little, and D. O'Shea. *Using algebraic geometry.* Springer-Verlag, New York, 1998.

[13] R. Dvornicich and C. Traverso. Newton symmetric functions and the arithmetic of algebraically closed fields. In *Applied algebra, algebraic algorithms and error-correcting codes (Menorca, 1987)*, pages 216–224. Springer, Berlin, 1989.

[14] M. Giusti, D. Lazard, and A. Valibouze. Algebraic transformations of polynomial equations, symmetric polynomials and elimination. In *Symbolic and algebraic computation (Rome, 1988)*, pages 309–314. Springer, Berlin, 1989.

[15] R. W. Gosper, Jr. Decision procedure for indefinite hypergeometric summation. *Proc. Nat. Acad. Sci. U.S.A.*, 75(1):40–42, 1978.

[16] R. Göttfert and H. Niederreiter. On the minimal polynomial of the product of linear recurring sequences. *Finite Fields Appl.*, 1(2):204–218, 1995. Special issue dedicated to Leonard Carlitz.

[17] P. Henrici. *Applied and computational complex analysis. Vol. 3.* John Wiley & Sons Inc., New York, 1986. Discrete Fourier analysis—Cauchy integrals—construction of conformal maps—univalent functions, A Wiley-Interscience Publication.

[18] R. Loos. Computing in algebraic extensions. In *Computer algebra*, pages 173–187. Springer, Vienna, 1983.

[19] P. Paule. Greatest factorial factorization and symbolic summation. *J. Symbolic Comput.*, 20(3):235–268, 1995.

[20] P. Penfield, Jr., R. Spence, and S. Duinker. *Tellegen's theorem and electrical networks.* The M.I.T. Press, Cambridge, Mass.-London, 1970.

[21] M. Petkovšek. Hypergeometric solutions of linear recurrences with polynomial coefficients. *J. Symbolic Comput.*, 14(2-3):243–264, 1992.

[22] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. Assoc. Comput. Mach.*, 27(4):701–717, 1980.

[23] V. Shoup. New algorithms for finding irreducible polynomials over finite fields. *Math. Comp.*, 54(189):435–447, 1990.

[24] V. Shoup. Fast construction of irreducible polynomials over finite fields. *J. Symbolic Comput.*, 17(5):371–391, 1994.

[25] V. Shoup. Efficient computation of minimal polynomials in algebraic extensions of finite fields. In *Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation (Vancouver, BC)*, pages 53–58, New York, 1999. ACM.

[26] J. von zur Gathen and J. Gerhard. Fast algorithms for Taylor shifts and certain difference equations. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation (Kihei, HI)*, pages 40–47 (electronic), New York, 1997. ACM.

[27] J. von zur Gathen and J. Gerhard. *Modern computer algebra.* Cambridge University Press, New York, 1999.

[28] K. Yokoyama, Z. Li, and I. Nemes. Finding roots of unity among quotients of the roots of an integral polynomial. In *Proceedings of the 1995 international symposium on Symbolic and algebraic computation*, pages 85–89. ACM Press, 1995.

Laboratoire Gage, École polytechnique, France and IMAR, Romania

E-mail address: Alin.Bostan@polytechnique.fr

Projet Algorithmes, Inria Rocquencourt, France

E-mail address: Philippe.Flajolet@inria.fr

Projet Algorithmes, Inria Rocquencourt, France

E-mail address: Bruno.Salvy@inria.fr

Laboratoire Gage, École polytechnique, France

E-mail address: Eric.Schost@polytechnique.fr

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399